Claims

What is claimed is:

1. A client side HTTP stack software component for processing requests, comprising:

at least one completion port object;

a thread pool comprising a plurality of threads adapted to process tasks associated with at least one client side request; and

a client side state machine associated with the at least one request.

2. The client side HTTP stack implementation of claim 1, further comprising a scheduler thread adapted to activate an object scheduled to begin sending requests at a specific time.

3. The client side HTTP stack implementation of claim 1, further comprising a DNS thread adapted to resolve domain names into IP addresses.

4. The client side HTTP stack implementation of claim 1, further comprising a timeout thread with a list of active sockets and timers associated with each socket, and adapted to selectively timeout at least one socket according to at least one timer in the list.

5. The client side HTTP stack implementation of claim 4, further comprising a scheduler thread adapted to activate an object scheduled to begin sending requests at a specific time.

6. The client side HTTP stack implementation of claim 5, further comprising a DNS thread adapted to resolve domain names into IP addresses.

7. The client side HTTP stack implementation of claim 4, further comprising a DNS thread adapted to resolve domain names into IP addresses.

8. A software component for implementing a client side HTTP stack, comprising:

a thread pool comprising N threads adapted to process M requests from a client application component, wherein N and M are integers greater than 1 and wherein M is greater than N.

9. The software component of claim 8, further comprising at least one thread activation component adapted to activate at least one of the N threads based on an event.

10. The software component of claim 9, wherein the at least one thread activation component is a completion port.

11. The software component of claim 9, wherein at least one of the N threads is adapted to deactivate itself and return to the thread pool when an operation being processed by the at least one of the threads is pending.

12. The software component of claim 11, wherein the event is the receipt of a completion packet by the at least one thread activation component.

13. The software component of claim 12, wherein the at least one thread activation component is a completion port.

14. The software component of claim 13, further comprising a scheduler thread adapted to activate an object scheduled to begin sending requests at a specific time.

15. The software component of claim 14, further comprising a DNS thread adapted to resolve domain names into IP addresses.

16. The software component of claim 15, further comprising a timeout thread with a list of active sockets and timers associated with each socket, and adapted to selectively timeout at least one socket according to at least one timer in the list.

17. The software component of claim 9, further comprising a state machine associated with at least one of the M requests.

18. The software component of claim 17, further comprising at least one key associated with the at least one of the M requests, wherein a first one of the N threads is associated with the at least one of the M requests, and wherein the thread activation component is adapted to associate the context of the first one of the N threads with the at least one state machine using the at least one key, in order to activate the first one of the N threads.

19. The software component of claim 18, wherein the thread activation component is adapted to associate the context of one of the N threads with the at least one state machine using the at least one key in order to activate the one of the N threads based on an event.

20. The software component of claim 8, further comprising a scheduler thread adapted to activate an object scheduled to begin sending requests at a specific time.

21. The software component of claim 8, further comprising a DNS thread adapted to resolve domain names into IP addresses.

22. The software component of claim 8, further comprising a timeout thread with a list of active sockets and timers associated with each socket, and adapted to selectively timeout at least one socket according to at least one timer in the list.

23. A method of implementing a client side HTTP stack, comprising:

processing M requests from a client application component using a thread pool comprising N threads, wherein M and N are integers greater than 1 and wherein M is greater than N.

24. The method of claim 23, further comprising:
selectively deactivating at least one of the N threads; and
activating at least another of the N threads based on an event using at least one thread activation component.

25. The method of claim 24, wherein the at least one thread activation component is a completion port.

26. The method of claim 24, wherein selectively deactivating at least one of the N threads comprises deactivating the at least one of the N threads when an operation being processed by the at least one of the N threads is pending.

27. The method of claim 26, wherein activating at least another of the N threads based on an event comprises:
receiving a completion packet using the thread activation component; and
activating one of the N threads upon receipt of the completion packet using the thread activation component.

28. The method of claim 27, wherein the at least one thread activation component is a completion port.

29. The method of claim 28, further comprising activating an object scheduled to begin sending requests at a specific time using a scheduler thread.

30. The method of claim 29, further comprising resolving domain names into IP addresses using a DNS thread.

31. The method of claim 30, further comprising selectively timing out at least one socket according to at least one timer associated with the at least one socket using a timeout thread comprising a list of active sockets and timers associated with each socket.

32. The method of claim 26, further comprising associating a state machine with at least one of the M requests.

33. The method of claim 32, further comprising:

associating at least one key with the at least one of the M requests;

associating a first one of the N threads with the at least one of the M requests; and

associating a context of the first one of the N threads with the at least one state machine using the at least one key, in order to deactivate the first one of the N threads.

34. The method of claim 33, further comprising associating a context of one of the N threads with the at least one state machine using the at least one key in order to activate the one of the N threads based on an event.

35. A computer-readable medium having computer-executable instructions for processing M requests from a client application component using a thread pool comprising N threads, wherein M and N are integers greater than 1 and wherein M is greater than N.

36. The computer-readable medium of claim 35, further comprising computer-executable instructions for:

selectively deactivating at least one of the N threads; and

activating at least another of the N threads based on an event using at least one thread activation component.

37. The computer-readable medium of claim 36, wherein the at least one thread activation component is a completion port.

38. The computer-readable medium of claim 36, wherein the computer-executable instructions for selectively deactivating at least one of the N threads comprises computer-executable instructions for deactivating the at least one of the N threads when an operation being processed by the at least one of the N threads is pending.

39. The computer-readable medium of claim 38, wherein the computer-executable instructions for activating at least another of the N threads based on an event comprises computer-executable instructions for:

receiving a completion packet using the thread activation component; and

activating one of the N threads upon receipt of the completion packet using the thread activation component.

40. The computer-readable medium of claim 39, further comprising computer-executable instructions for activating an object scheduled to begin sending requests at a specific time using a scheduler thread.

41. The computer-readable medium of claim 40, further comprising computer-executable instructions for resolving domain names into IP addresses using a DNS thread.

42. The computer-readable medium of claim 41, further comprising computer-executable instructions for selectively timing out at least one socket according to at least one timer associated with the at least one socket using a timeout thread comprising a list of active sockets and timers associated with each socket.

43. The computer-readable medium of claim 43, further comprising computer-executable instructions for associating a state machine with at least one of the M requests.

44. The computer-readable medium of claim 43, further comprising computer-executable instructions for:

associating at least one key with the at least one of the M requests;

associating a first one of the N threads with the at least one of the M requests; and

associating a context of the first one of the N threads with the at least one state machine using the at least one key, in order to deactivate the first one of the N threads.

45. The computer-readable medium of claim 44, further comprising computer-executable instructions for associating a context of one of the N threads with the at least one state machine using the at least one key in order to activate the one of the N threads based on an event.

46. A software component for implementing a client side HTTP stack, comprising:

means for processing M requests from a client application component using a thread pool comprising N threads, wherein M and N are integers greater than 1 and wherein M is greater than N.

47. The software component of claim 46, further comprising:

means for selectively deactivating at least one of the N threads; and

means for activating at least another of the N threads based on an event.

48. The software component of claim 47, further comprising means for activating an object scheduled to begin sending requests at a specific time.

49. The software component of claim 47, further comprising means for resolving domain names into IP addresses.

50. The software component of claim 47, further comprising means for selectively timing out at least one socket according to at least one timer associated with the at least one socket.